DL4CV - Assignment Report
Non-Astemici

Filippo De Min, Luca D'Ambrosio,

3143769 3121995

December 5, 2024

1 Introduction

Buildings are what characterizes a city's landscape, as well as how people perceive it and live it. From a urbanist perspective, the age of a building conveys a lot of information, such as if the building might need renovation, or if there's sismic risk; from an historian perspective, walking around a city knowing when each building has been constructed, adds value to visiting such city.
We propose a methodology to **predict** a Lombardy based buildings' **year of construction**, while simultaneously **classifying** its broad **typology**, based on the image of its facade.
We scrape data from *regionelombardia.it*, collecting all reported images of buildings, as well as their age, and typology, and build a dataset of roughly 9000 images.
Next, we employ a **transfer learning** framework, where we process our images through **ResNet50**[1] and build a *multitask* top level multi layer perceptron consisting of two separate output heads: a regression head (year of construction) and a classification head (general typology).

Our methodology consists of two main steps

- ◇ **train** our MLP by *freezing* the pre-trained convolutional layers.
- ◇ **fine-tune** the convolutional layers by *freezing* the MLP layers.

We contribute to the literature with a novel dataset and methodology for tackling this task.

2 Data

2.1 Data Collection

The dataset for this analysis was extracted from Lombardia Beni Culturali, the regional portal dedicated to cultural heritage in Lombardia. This platform provides detailed information about buildings of cultural and historical significance across the region. To collect the data, we employed the Selenium Python library to develop an automatic web scraper allowing us to collect all the 15,318 observations. Each building in the dataset was assigned a unique identifier corresponding to its page URL and the images were saved locally. We decided to focus our analysis on *General Typology* and *Construction Date* to provide both a classification and regression setting.

2.2 Processing

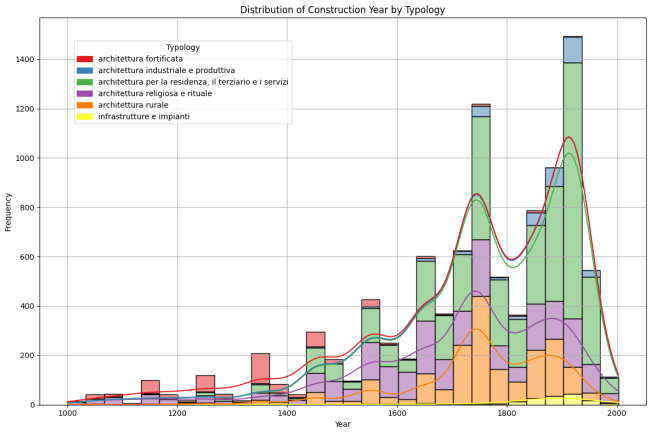

Figure 1: Distribution of the observations across date and category

Filtering out observations for which the date was missing, we are left with 10,810. Processing the date was challenging as data was textual. We transformed roman centuries date (XV cent.) to integers. Century' qualifiers like *"early" XV* were replaced with 20 and hence became 1420. When 2 dates were provided, indicating a range of possible construction dates, we used the average. We did not lose any observation in this process. Finally, we decided to not

include constructions for which the range of dates was greater than 200 years suggesting too much uncertainty. We also filtered out buildings constructed prior to year 1500 due to little representativeness. We are left with a final dataset of 9814 observations. The train set is 80% of the data, of the remaining 20%, 90% constitute the validation set and the rest is the test set. Splits are stratified by typology.
As shown in Figure 1 construction year is not equally distributed among categories. This motivates a strategy of two separate output heads in which the model can learn different patterns from the FEN.

3 Methods

3.1 CNN

For this project, we opted for a transfer learning approach, as we believe it would yield better results for our task.
In particular, we employ the convolutional block of ResNet50, due to its proven ability in feature extraction, while not resorting to a larger model (e.g. ResNet-101 or ResNet-152) due to computational constraints.
We then connect our own MLP to the output of the feature extraction network. The network can be described as follows:

- ◇ **FEN**: ResNet50 convolutional block, it outputs a tensor of $shape = (7, 7, 2048)$
- ◇ **Flatten**: We decided to flatten the FEN output as it gave better results compared to global average pooling. This is likely due to the images containing a lot of noise (some photos were taken from far away), and a pooling approach might delete some details that are just a few pixels wide, althouth they are necessary for the task (e.g. if an image was taken from far away, with GAP the network might not pick up on the type of window installed).
- ◇ **Batch Normalization**: Due to the large amount of parameters, we deem appropriate the use of such a technique to stabilize the training.
- ◇ **Drop Out**: Again, due to the large amount of parameters, we include drop out to prevent overfitting.
- ◇ **Dense Layers**: Following the previous layers, the network is split in two sections: classification and regression. Both sections contain 2 hidden layers of respective size 256 and 128, adding batch normalization and drop out in between layers. The first head (classification) has a softmax activated output for 6 classes, while the second (regression) is linearly activated with 1 output neuron.

Furthermore, we chose RMSprop as an optimizer as it seems to work better with noisy data, with a larger learning rate in the training phase compared to the fine-tuning phase. The batch sizes are 64 for the training phase, but it had to be decreased to 16 in the fine-tuning phase due to RAM constraints. Moreover, in order to further prevent overfitting, we introduced early stopping in both phases, with 20 epochs of patience for the first phase and 10 for the second. Finally, only for the training phase, we set a Learning rate decrease to 1e-6 on plateau with patience set at 15 epochs. A summary of the hyperparameters used is reported in Table 2 (appendix).

3.2 Gradient Class Activation Maps

After training and evaluating the network and evaluating it on our test set, we implement GradCAM on our images to grasp which subsets of the image activate the most at inference time.
We implement a function that takes as parameters the image array (properly preprocessed) and the task for which we want to create a heatmap. We then compute the gradients:

$$G = \frac{\partial y_{class/reg}}{\partial A}, \quad G \in R^{u \times v \times K}.$$

$$(u, v, K) = (height, width, channels)$$

with respect to the appropriate output layer ("classes" in case of classification and "regression" in case of regression). These are then pooled across each channel, and pre multiplied by ResNet50's last convolutional layer (*conv5_block3_out*) to get an activation map for that specific task.

4 Results

4.1 CNN

Regarding the results of our analysis (Table 1), we are fairly disappointed in the outcome. Al-

Table 1: Validation and Test Metrics

Model	Validation					Test				
	Accuracy	Precision	Recall	F1	MAE	Accuracy	Precision	Recall	F1	MAE
FineTuned	0.77	0.51	0.45	0.47	83.9	0.80	0.57	0.52	0.53	87.1
Trained	0.76	0.51	0.44	0.46	91.2	0.76	0.54	0.46	0.48	92.4

Note: Mean Average Error is expressed in *years*.

though the accuracy metric seems solid, this result stems from the large class imbalance present in our dataset, which leads to the model classifying the majority of observations as residential buildings (largest proportion). This becomes clearer by looking at the other classification metrics. Although both models fail to identify the least prevalent classes, it still does a good job differentiating between: *residential*, *rural*, and *religious* with the majority of observations being classified correctly.
When it comes to the regression task, we can see that the average error is around 80 to 90 years, which is quite disappointing given that the architectural style is known to shift much quicker, with buildings built in the 1960s being drastically different than those built in the 1980s.
However, there seems to be an improvement when fine-tuning the model which are more evident in the test set metrics. In fact, we observe a 5 years reduction in the MAE and a 5 percentage points increase in F1 score. Moreover, test metrics are fairly similar to the validation metrics, which leads us to think that we successfully avoided overfitting.

4.2 Gradient Class Activation Map

Continuing our analysis by inspecting Grad Cam, we notice that the model struggles when the photograph has been taken from afar, or if there are other elements in the picture other than the building. In the cases in which the building is centered however, we notice that the focus resides on structural elements such as its windows.
By looking at the regression setting however, the model struggles more. We were expecting to observe this as the regression setting does not leverage class specific patterns but broader information in the picture. Moreover, gradients show sharper activations in the classification setting due to the nature of the loss function. 2 displays a correctly classified sample.

Figure 2: Grad-CAM for correctly classified sample

5 Discussion and Conclusion

We believe our analysis showed that this task is indeed possible, but there are some inherent limitations to our approach. Firstly, our dataset in quite unbalanced in terms of classes, and this is reflected in the poor F1 scores. Additionally, the general typology (residential, religious etc.) of the building might not be the correct target for this task, and using the specific typology (church, house, tower etc.) might be the more correct approach; this however would require a much larger dataset containing more samples for each class. The second limitation stems from the nature of the images: in fact, many of them include a lot of additional elements (trees, other houses etc.) which add noise and make both tasks quite unrealistic. Further research should consider ad-hoc cropping or using street view data to achieve better results.
Moreover, a multitask model might also not be the optimal strategy, and other architectures should be explored. Concluding, we are satisfied with the results of our work, although the outcome was not what we hoped for.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

A Hyperparameters

Table 2: CNN hyperparameters

Hyper-parameter	Training	Fine-Tuning
Loss (C)	Cross Entropy	Cross Entropy
Loss (R)	MSE	MSE
Optimizer	RMSprop	RMSprop
learning rate	0.005	2e-5
Non-Linearity	ReLu	ReLu
E.S. patience	20	10
L.R. plateau	15	x
batch size	64	16

Note: E.S = Early Stopping, L.R. = learning rate

B Model Architecture